Logistic Markov Decision Processes

Martin Mladenov[*]
TU-Dortmund
martin.mladenov@cs.tu-dortmund.de

Craig Boutilier
Google Research
cboutilier@google.com

Dale Schuurmans[†]
University of Alberta
daes@ualberta.ca

Ofer Meshi
Google Research
meshi@google.com

Gal Elidan
Google Research
elidan@google.com

Tyler Lu
Google Research
tylerlu@google.com

Abstract

User modeling in advertising and recommendation has typically focused on *myopic predictors* of user responses. In this work, we consider the *long-term decision problem* associated with user interaction. We propose a concise specification of long-term interaction dynamics by combining factored *dynamic Bayesian networks* with logistic predictors of user responses, allowing state-of-the-art prediction models to be seamlessly extended. We show how to solve such models at scale by providing a constraint generation approach for *approximate linear programming* that overcomes the variable coupling and non-linearity induced by the logistic regression predictor. The efficacy of the approach is demonstrated on advertising domains with up to 2^{54} states and 2^{39} actions.

1 Introduction

Online and mobile interaction with users has been transformed by the use of machine learning (ML) to predict user intent, preferences and responses. However, current methods largely focus on *myopic* prediction—predicting a user's immediate response to the system's action—without explicitly modeling long-term impact nor addressing the ultimate need to plan *sequences* of interactions with a user. For example, ad systems typically determine which ads to show based on a current prediction of a user's click probability. Evidence is mounting that myopic predictors compromise user satisfaction and system performance by ignoring long-term value [Hohnhold *et al.*, 2015]. While sequential models of user interaction using *Markov decision processes (MDPs)* and *reinforcement learning (RL)* have received increasing attention [Charikar *et al.*, 1999; Li *et al.*, 2009; Archak *et al.*, 2010; 2012; Amin *et al.*, 2012; Silver *et al.*, 2013; Theocharous *et al.*, 2015], they have yet to find widespread adoption due to challenges with modeling and computational scalability.

In this work, we develop a practical approach to optimizing long-term interactions with users by introducing the *logistic MDP*, a novel form of MDP that incorporates a logistic predictor of a user response variable (such as click probability) into an MDP. Two key insights motivate the development of logistic MDPs. First, the features used in (myopic) logistic predictors of user responses typically provide a sufficient statistic of user history well suited to (some forms of) long-term modeling. Second, the evolution of the features used in such models often exhibits considerable conditional independence, allowing for compact expression of a transition function as a *dynamic Bayesian network (DBN)* [Dean and Kanazawa, 1989; Boutilier *et al.*, 1999] that incorporates the predictor into the dynamics.

Once logistic MDPs are motivated and defined, we turn our attention to effectively solving such MDPs in a scalable fashion. Two key obstacles are created by the logistic user response model when devising algorithms that exploit the conditional independence in the dynamics: (i) variable coupling, and (ii) non-linearity introduced. We develop an *approximate linear programming (ALP)* solution method [Guestrin *et al.*, 2003; de Farias and Van Roy, 2003] that overcomes these obstacles. We do so by developing *constraint generation* strategies [Schuurmans and Patrascu, 2001] that search for violated constraints using a set of Boolean optimization problems (BOPs) defined over approximations of the logistic function. We provide exact and approximate ALP algorithms for logistic MDPs, and derive error bounds for the approximate algorithms. We demonstrate the effectiveness of the approach on large models, derived from in-app display advertising data, with up to 2^{54} states and 2^{39} actions.

2 Background

The prediction of user responses to recommendations, ads, or other interventions, are routinely tackled using ML. Such predictions are often binary (e.g., will an ad be clicked or an app downloaded) with the predicted probability of response—used to score potential actions (e.g., which ad to show)—often combined with other factors, such as an advertiser's bid. Many models have been used for such prediction, ranging from logistic regression [Richardson *et al.*, 2007; Graepel *et al.*, 2010; McMahan *et al.*, 2013] to deep models [Covington *et al.*, 2016; Cheng *et al.*, 2016].

Since linear logistic regression is the current workhorse for predicting user responses to online ads, we focus on these models in this work.[1] We assume a finite set of *variables*

[*]Work performed while author was a visiting intern at Google.
[†]Work performed while author was a visiting scholar at Google.

[1]We briefly discuss extensions of our approach to deep neural network models (DNNs) in the concluding section.

$\mathcal{X} = \{X_1, \ldots, X_n\}$ describing properties of the interaction (e.g., geolocation, age, purchase activity, device), ad (e.g., advertiser, product type), context (e.g., site, app), and user history (e.g., count of past ad impressions, clicks). As is typical in practice, we also assume each X_i has a discrete domain $Dom(X_i)$ and that a "one-hot" encoding is used to transform elements of $Dom(\mathcal{X})$ into a sparse binarized *feature vector* of length $\sum_i |Dom(X_i)|$. Let φ be a binary *user response variable* (e.g., user click on an ad). Given a learned weight vector $\mathbf{u}$, a *linear logistic predictor* models the probability of a positive user response to a given feature vector $\mathbf{x}$ as:

$$P(\varphi = \top|\mathbf{x}) = \sigma(\mathbf{u}^{\mathrm{T}}\mathbf{x}) = \frac{1}{1 + e^{-\mathbf{u}^{\mathrm{T}}\mathbf{x}}}. \quad (1)$$

In ad settings, we refer to this as a *pCTR model*.

Sequential models of user behavior in advertising have been studied [Charikar *et al.*, 1999; Archak *et al.*, 2010; Li *et al.*, 2009], though not extensively in practice. Recently, MDP models for optimizing ad serving and promotions have been proposed [Archak *et al.*, 2012; Silver *et al.*, 2013; Theocharous *et al.*, 2015], and temporal models for recommender systems have also received some attention [Shani *et al.*, 2005; Taghipour *et al.*, 2007; Rendle *et al.*, 2010; He and McAuley, 2016; Sahoo *et al.*, 2012; Tan *et al.*, 2016; Wu *et al.*, 2017]. Unfortunately, these methods have yet to find their way into practical recommender and ad serving systems, since realistic models of user dynamics are difficult to learn or specify in practice, and computing optimal behaviour policies in such models is computationally complex, whether by explicitly solving an MDP or via reinforcement learning.

3 Logistic MDPs

To overcome the challenges described above, we introduce a new MDP model, *logistic MDPs*, that extends standard MDPs by introducing a logistic user response variable.

3.1 Non-myopic Models of User Interactions

As noted above, most systems for ad serving or content/product recommendations are *myopic* in two distinct senses. First, they predict a user's response to a system action (e.g., ad served) without considering the action's impact on subsequent interactions. Second, the prediction is often for some immediate user response (e.g., pCTR) rather than long-term behavior (cumulative click rate, user satisfaction, etc.). This is problematic for several reasons. Intuitively, system actions at a given time will also influence user responses to *future* actions; e.g. display ad exposure may increase a user's propensity to click on the related ads by increasing awareness through exposure, or may decrease this propensity by creating a "blindness" effect [Hohnhold *et al.*, 2015]. Additionally, an action and user response usually changes the user feature vector used for the *next* prediction, hence influences the predicted effectiveness of all subsequent actions.

Since the aim is not to optimize a single user interaction, but to do so over (say) a session, week, or lifetime, it is natural to model this engagement with an MDP. One can then construct an optimal policy by directly solving the MDP [Archak *et al.*, 2012; Amin *et al.*, 2012; Boutilier and Lu, 2016], or applying RL to interaction data [Silver *et al.*, 2013; Theocharous *et al.*, 2015].

In this work, we focus on solving an MDP model directly. One reason is that long-term optimization often induces tradeoffs among different objectives that should be encoded in the reward function.[2] Quantifying these tradeoffs precisely in a reward function can be difficult *a priori*. Model-based methods allow decision makers to efficiently explore the tradeoffs arising from different reward functions. In addition to being used directly, model-bsaed solutions computed offline can be used to accelerate (online or offline) RL.

However, such an approach requires a transition model that captures the dynamics of user behavior (expected user actions, evolution of user state, user responses to actions, etc.). A key challenge is specifying the *state space*; i.e., defining an appropriate set of features, observable from logged interactions, that adequately summarizes user history in a way that is both predictive of future behavior and renders the implied dynamics Markovian.

Constructing a Model of Interaction Dynamics Fortunately, one does not need to start from scratch. Instead, we *leverage the existing myopic predictors* of user response as follows. Focusing on logistic regression, a typical model is illustrated in Fig. 1(a), where the response variable φ (e.g., representing a positive ad click) depends on the variables X_i representing the user state, user history and relevant context (e.g., web site visited, query issued, app being used, etc.), and variables A_i reflecting various properties of the system action (e.g., ad being shown, app or video being recommended). If we seek to optimize the clicks accumulated during a user session, we must predict not only how actions influence the immediate response, but how they *change the variable values we expect to encounter subsequently*. Fig. 1(b) illustrates how the myopic response model can be extended to accommodate such dynamics, where we view the variables X_i as *state variables* and the variables A_i as *action variables*. The model shows how the state variables at time t+1 depend (probabilistically) on the state at time t, the action taken at time t, and the user's response to that action.

This provides a natural, practical way to construct the required MDP model of user dynamics, which exploits two key characteristics of typical (myopic) response models. First, the features in the myopic model provide a suitable summary of history (i.e., form a sufficient statistic) for predicting response variable φ. Second, the features themselves are (roughly) "self-predictive;" i.e., the features at time t (together with φ) are often sufficient to predict each state variable X_i. In many practical models, this assumption holds (e.g., some variables reflect static user properties, others capture summary statistics of past user behavior, while still others model the natural evolution of user actions). As a result, one can readily develop models reflecting the evolution of these variables from historical interaction data (assuming sufficient "explicit" exploration or other induced randomness)—indeed, the same logged data used to construct myopic response models can be used directly for this purpose. Finally, the prediction of (the

[2] For instance, in *app recommendations*, install rate, user app engagement (post-install), revenue generation potential for the developer, and many other factors may play a role.

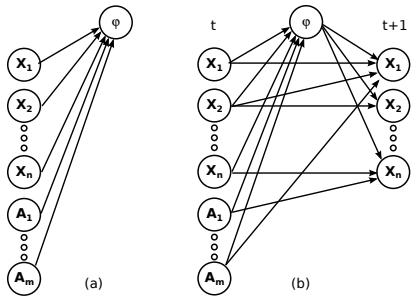

Figure 1: An illustration of (a) a (myopic) logistic regression response model, and (b) the dependence structure of a logistic MDP that extends the myopic response model.

distribution of) state variables X_i at time $t+1$ often depends on only a small number of preceding state/action features and (sometimes) the response variable.

3.2 Logistic MDPs: Formal Model

We formalize these intuitions in the *logistic MDP* model, a form of *factored MDP* using a typical DBN representation [Boutilier *et al.*, 1999] in which the DBN is augmented with the response variable φ.

Markov Decision Processes A (finite) MDP [Puterman, 1994] is given by: a finite state space S and action space A; a stochastic transition model P, with $P(s,a,s') = p^a_{ss'}$ denoting the probability of a transition from state s to s' when action a is taken; and a bounded, real-valued reward function $R(s,a) = r^a_s$ denoting the immediate (expected) reward of taking action a at state s. A *(stationary, deterministic) policy* π specifies an action $\pi(s) \in A$ to be taken in any state s. We seek an optimal policy that maximizes the expected sum of discounted rewards (with discount factor $0 \leq \gamma < 1$). The (unique) *optimal value function* $V^* : S \to \mathbb{R}$ satisfies the Bellman equation (2), and induces the *optimal policy* (3), for all $s \in S$:

$$V^*(s) = \max_{a \in A} r^a_s + \gamma \sum_{s' \in S} p^a_{ss'} V^*(s'), \tag{2}$$

$$\pi^*(s) = \operatorname*{argmax}_{a \in A} r^a_s + \gamma \sum_{s' \in S} p^a_{ss'} V^*(s'). \tag{3}$$

Given an arbitrary (not necessarily optimal) *value function (VF)* $V : S \to \mathbb{R}$, the *action-value backup* is:

$$Q^V(s,a) = r^a_s + \gamma \sum_{s' \in S} p^a_{ss'} V(s'). \tag{4}$$

In this work we use *linear programming (LP)* to solve MDPs [Puterman, 1994]. For a finite MDP, the primal LP formulation is:

$$\min_{v_s : s \in S} \quad \sum_{s \in S} \alpha_s v_s \tag{5}$$

$$\text{s.t.} \quad v_s \geq Q^v(s,a), \forall s \in S, a \in A, \tag{6}$$

where variables v_s reflect the optimal VF at each state s, $Q^v(\cdot, a)$ is the action-value backup (treating variable vector v as a VF), and state weighting function α is usually interpreted as the initial state distribution.[3]

Factored MDPs In realistic settings, MDPs are specified using finite sets of state and action variables, $\mathcal{X} = \{X_1, \ldots, X_n\}$ and $\mathcal{A} = \{A_1, \ldots, A_m\}$, respectively. Each state variable X_i has finite domain $Dom(X_i)$ (similarly for action variables). Since the induced state and action spaces, $S = Dom(\mathcal{X})$ and $A = Dom(\mathcal{A})$, have exponential size, DBNs are often used to compactly represent system dynamics [Dean and Kanazawa, 1989; Boutilier *et al.*, 1995; 1999]. Two assumptions allow for the compact representation of the successor state distributions $\Pr(\mathbf{x}^{(t+1)}|\mathbf{x}^{(t)}, \mathbf{a}^{(t)})$ using a standard DBN (ignoring φ for now): (i) they are products of marginals over the individual state variables;[4] and (ii) each variable X_i's local distribution is dependent on a small *parent set* $Par_i \subset \mathcal{X} \cup \mathcal{A}$. This means that the local distributions themselves can be specified compactly.

Logistic MDPs We augment the standard DBN representation with the binary response variable as follows.[5] We assume the state-action pair $(\mathbf{x}^{(t)}, \mathbf{a}^{(t)})$ at time t determines $\Pr(\varphi^{(t+1)})$, and the triple $(\mathbf{x}^{(t)}, \mathbf{a}^{(t)}, \varphi^{(t+1)})$ dictates the next state distribution $\Pr(\mathbf{x}^{(t+1)})$. Hence the dynamics factor as:

$$\Pr(\mathbf{x}^{(t+1)}|\mathbf{x}^{(t)}, \mathbf{a}^{(t)}) = \textstyle\sum_{\varphi \in \{\top, \bot\}} \Pr(\mathbf{x}^{(t+1)}|\mathbf{x}^{(t)}, \mathbf{a}^{(t)}, \varphi^{(t+1)}) \Pr(\varphi^{(t+1)}|\mathbf{x}^{(t)}, \mathbf{a}^{(t)}). \tag{7}$$

We assume $\Pr(\mathbf{x}^{(t+1)}|\mathbf{x}^{(t)}, \mathbf{a}^{(t)}, \varphi^{(t+1)})$ is a product distribution over the $t+1$ state variables, with $\Pr(X_j^{(t+1)}|Par_j^{(t)}, \varphi^{(t+1)})$ depending on a small set of parents.[6] $\Pr(\varphi^{(t+1)}|\mathbf{x}^{(t)}, \mathbf{a}^{(t)})$ is given by the logistic model in (1). Fig. 1(b) illustrates the dependence structure.

The reward function is also assumed to be factored, decomposing as the sum of r *reward factors* ρ_i:

$$R(\mathbf{x}, \mathbf{a}, \varphi) = \textstyle\sum_{i \leq r} \rho_i(\mathbf{x}[R_i], \mathbf{a}[R_i], \varphi), \tag{8}$$

where each $R_i \subset \mathcal{X} \cup \mathcal{A}$ is small set of state/action variables, such that $\mathbf{x}[U]$ (resp., $\mathbf{a}[U]$) denotes the restriction of a variable instantiation to the subset $U \subseteq \mathcal{X} \cup \mathcal{A}$. Unlike traditional factored MDPs, reward factors may also depend on φ.

This transition model is very compact, with the local distribution for each X_i having $O(|Dom(Par_i)||Dom(X_i)|)$ parameters, while φ requires $O(\sum_i Dom(X_i))$ parameters. However, as shown in Fig. 1(b), unlike a typical DBN, *φ correlates all of the $t+1$ variables to which it is connected—destroying the structure that is normally exploited in DBN MDP algorithms*. We address this issue next.

4 ALP for Logistic MDPs

The reward and transitions structure of the usual DBN representation of an MDP can be exploited by various solution algorithms [Boutilier *et al.*, 1995; Hoey *et al.*, 1999; St-Aubin *et al.*, 2000; Boutilier *et al.*, 1999]. In this work,

[3] Any strictly positive vector α suffices. The role of the state weighting is less straightforward in approximate linear programming (next section) [de Farias and Van Roy, 2003].

[4] Our approaches also allow correlations among post-action variables, but to simplify notation, we assume full independence (conditional on the prior state and action).

[5] Technically, the resulting model remains a DBN, but since the response variable destroys much of the structure typically assumed in a DBN, we treat it differently from other variables.

[6] Any specific X_j may be independent of φ, but if not, we sometimes refer to φ as a parent of X_j.

we focus on *approximate linear programming (ALP)* for factored MDPs [Guestrin *et al.*, 2003; Schuurmans and Patrascu, 2001], where the value function is approximated using a linear combination of *basis functions* or "features" $\mathcal{B} = \{\beta_1, \ldots, \beta_k\}$ over $\mathcal{X}$. Here, each β_i is a function of some small set $B_i \subset \mathcal{X}$, where we assume a *bias factor* $\beta_{bias} = \mathbf{1} \in \mathcal{B}$. The resulting value function (VF) is parameterized by a weight vector $\mathbf{w}$:

$$V(\mathbf{x}; \mathbf{w}) = \sum_{i \leq k} w_i \beta_i(\mathbf{x}[B_i]) = \mathbf{B_x w}, \tag{9}$$

where $\mathbf{B} = [\beta_1, \ldots, \beta_k]$ is the *basis matrix*, and $\mathbf{B_x}$ denotes the row of $\mathbf{B}$ corresponding to state $\mathbf{x}$. When the weighting function α is factored similarly, the LP becomes:[7]

$$\min_{\mathbf{w}} \quad \sum_{i \leq k} \sum_{\mathbf{b}_i \in Dom(B_i)} \alpha[\mathbf{b}_i] w_i \beta_i(\mathbf{b}_i) \tag{10}$$

$$\textit{s.t.} \quad 0 \geq h(\mathbf{x}, \mathbf{a}; \mathbf{w}), \forall \mathbf{x}, \mathbf{a}, \tag{11}$$

where $h(\mathbf{x}, \mathbf{a}; \mathbf{w}) = Q^V(\mathbf{x}, \mathbf{a}; \mathbf{w}) - V(\mathbf{x}; \mathbf{w})$. Note that the constraint for each state-action pair has a compact representation, since $V(\mathbf{x}; \mathbf{w})$ is given by (9), and $Q^V(\mathbf{x}, \mathbf{a}; \mathbf{w})$ is a reward term plus an expectation of a linear combination of basis functions. For each β_i, denote its backprojection by:

$$\begin{aligned} g_i(\mathbf{x}, \mathbf{a}) &= g_i(\mathbf{x}[Par_{B_i}], \mathbf{a}[Par_{B_i}]) \\ &= \sum_{\mathbf{y} \in Dom(B_i)} \Pr(\mathbf{y} | \mathbf{x}[Par_{B_i}], \mathbf{a}[Par_{B_i}]) \beta_i(\mathbf{y}). \end{aligned} \tag{12}$$

Hence β_i depends only on variables in $Par_{B_i} = \cup_{X_j \in B_i} Par_j$ [Boutilier *et al.*, 1995; Guestrin *et al.*, 2003]. V and Q^V can be combined to yield

$$\begin{aligned} h(\mathbf{x}, \mathbf{a}; \mathbf{w}) &= \sum_{j \leq r} \rho_j(\mathbf{x}[R_j], \mathbf{a}[R_j]) \\ &+ \sum_{i \leq k} w_i \Big(\gamma \cdot g_i(\mathbf{x}[Par_{B_i}], \mathbf{a}[Par_{B_i}]) - \beta_i(\mathbf{x}[B_i]) \Big), \end{aligned} \tag{13}$$

so the LP has a concise objective and constraints.

To handle the exponential number of constraints, we use *ALP with constraint generation (ALPCG)* [Schuurmans and Patrascu, 2001], where a relaxed LP (the *master*) is repeatedly solved using a subset of the constraints $C \subseteq (\mathcal{X}, \mathcal{A})$. In particular, given the solution $\mathbf{w}$ to the master LP, detecting the *maximally violated constraint (MVC)* is a *Boolean optimization problem (BOP)*:

$$\begin{aligned} \max_{\mathbf{x}, \mathbf{a}} \quad & \sum_{j \leq r} \rho_j(\mathbf{x}[R_j], \mathbf{a}[R_j]) \\ & + \sum_{i \leq k} w_i \Big(g_i(\mathbf{x}[Par_{B_i}], \mathbf{a}[Par_{B_i}]) - \beta_i(\mathbf{x}[B_i]) \Big). \end{aligned} \tag{14}$$

The subproblem uses Boolean (indicator) variables to express the assignment of state and action variables in $\mathbf{x}$ and $\mathbf{a}$, as well as the selection of terms in the sub-functions g_i, β_i, ρ_j, with constraints enforcing the consistency of the assignments. The BOP can be solved using a standard Boolean or mixed integer solver. At each iteration the MVC is added to C, tightening the relaxation, and the master LP is re-solved. If no violated constraint exists, the current solution is optimal.

This approach can be very effective for MDPs with compact DBN representations [Schuurmans and Patrascu, 2001], and by terminating ALPCG early, one can further approximate the solution of the MDP.

[7] Any compact factorization of the weighting function suffices—we assume this specific form for ease of exposition.

ALPCG for Logistic MDPs To extend ALPCG to logistic MDPs we need to overcome two complications in the constraint generation subproblem: the loss of conditional independence among state variables, and the non-linearity of the response model in the state-action space.

Using the VF representation (9), the action-value backup now has the form:

$$\begin{aligned} Q^V(\mathbf{x}, \mathbf{a}; \mathbf{w}) = \sum_{\varphi \in \{\top, \bot\}} \Pr(\varphi | \mathbf{x}, \mathbf{a}) \Big[& \sum_{j \leq r} \rho_j(\mathbf{x}[R_j], \mathbf{a}[R_j], \varphi) \\ & + \gamma \sum_{i \leq k} w_i g_i(\mathbf{x}[Par_{B_i}], \mathbf{a}[Par_{B_i}], \varphi) \Big], \end{aligned} \tag{15}$$

where the backprojection g_i for basis function β_i is defined as in (12), but where we *fix* φ and then take an expectation w.r.t. its realizations ($\top$ or $\bot$). The same *conditioning* applies to the reward component ρ_j of the action-value backup. Thus, the constraint function h can be re-expressed as

$$\begin{aligned} h(\mathbf{x}, \mathbf{a}, \varphi; \mathbf{w}) &= \sum_{j \leq r} \rho_j(\mathbf{x}[R_j], \mathbf{a}[R_j], \varphi) \\ &+ \sum_{i \leq k} w_i \Big(\gamma \cdot g_i(\mathbf{x}[Par_{B_i}], \mathbf{a}[Par_{B_i}], \varphi) - \beta_i(\mathbf{x}[B_i]) \Big). \end{aligned} \tag{16}$$

An ALP for the logistic MDP can therefore be expressed with the same objective (10) as before, but with the constraint:

$$0 \geq C(\mathbf{x}, \mathbf{a}; \mathbf{w}) \quad \forall \mathbf{x}, \mathbf{a}, \quad \text{where} \tag{17}$$

$$C(\mathbf{x}, \mathbf{a}; \mathbf{w}) = \sum_{\varphi \in \{\top, \bot\}} \Pr(\varphi | \mathbf{x}, \mathbf{a}) h(\mathbf{x}, \mathbf{a}, \varphi; \mathbf{w}). \tag{18}$$

This LP has the same desirable features as the ALP formulation for standard DBNs: a compact objective and compact constraints. As with standard ALPCG, we handle the exponential number of constraints using constraint generation.

Search for Maximally Violated Constraints We now turn to the constraint generation problem. Assuming the ALP above has been solved using only a *subset* of the constraints (17), we wish to find an MVC at the current solution:

$$\max_{\mathbf{x}, \mathbf{a}} \quad \sum_{\varphi \in \{\top, \bot\}} \Pr(\varphi | \mathbf{x}, \mathbf{a}) \cdot h(\mathbf{x}, \mathbf{a}, \varphi; \mathbf{w}). \tag{19}$$

Let $f(\mathbf{x}, \mathbf{a})$ denote the linear function of state-action features that is passed through the sigmoid in the logistic regression response model (1); that is, $\Pr(\varphi = \top | \mathbf{x}, \mathbf{a}) = \sigma(f(\mathbf{x}, \mathbf{a}))$. For any interval $[f_\ell, f_u]$ of the input space to the sigmoid, we (ab)use the same notation to denote the set of states: $[f_\ell, f_u] = \{(\mathbf{x}, \mathbf{a}) : f_\ell \leq f(\mathbf{x}, \mathbf{a}) \leq f_u\}$.

We now define a search procedure to solve (19). Our search for an MVC proceeds by recursively searching in intervals $[f_\ell, f_u]$ for an MVC among pairs $(\mathbf{x}, \mathbf{a}) \in [f_\ell, f_u]$. We call the ALP procedure using this search *ALP-SEARCH*.

The search for an MVC in interval $[f_\ell, f_u]$ is complicated by the non-linearity of the constraint function $C(\mathbf{x}, \mathbf{a}; \mathbf{w})$ in (18). However, if we replace $\Pr(\varphi | \mathbf{x}, \mathbf{a})$ by some constant σ_* for all $(\mathbf{x}, \mathbf{a}) \in [f_\ell, f_u]$, then the optimization (19) to find an MVC over the interval becomes linear:

$$\max_{\mathbf{x}, \mathbf{a}} \quad \sigma_* h(\mathbf{x}, \mathbf{a}, \top; \mathbf{w}) + (1 - \sigma_*) h(\mathbf{x}, \mathbf{a}, \bot; \mathbf{w}) \tag{20}$$

$$\textit{s.t.} \quad f_\ell \leq f(\mathbf{x}, \mathbf{a}) \leq f_u. \tag{21}$$

Since $\sigma(z)$ is monotonically increasing in z, the quantity $\sigma_u = \sigma(f_u)$ provides an upper bound on $\Pr(\varphi | \mathbf{x}, \mathbf{a})$ for any $(\mathbf{x}, \mathbf{a}) \in [f_\ell, f_u]$ (see Fig. 2 for an illustration).

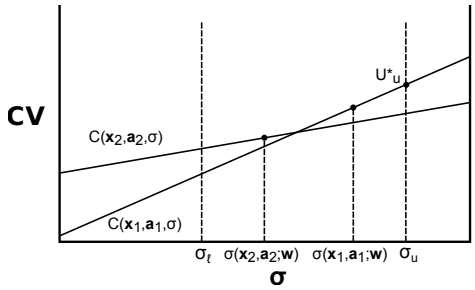

Figure 2: MVC search example: Both pairs $(\mathbf{x}_1, \mathbf{a}_1), (\mathbf{x}_2, \mathbf{a}_2) \in H^+_{\mathbf{w}}$ lie in interval $[f_\ell, f_u]$ (in probability space, $[\sigma_\ell, \sigma_u]$). When $\sigma = \sigma_u$, $(\mathbf{x}_1, \mathbf{a}_1)$ induces upper bound U^*_u on constraint violation among *all* pairs in $H^+_{\mathbf{w}} \cap [f_\ell, f_u]$. Since $(\mathbf{x}_1, \mathbf{a}_1)$'s true probability $\sigma(f(\mathbf{x}_1, \mathbf{a}_1; \mathbf{w}))$ is less than σ_u, the bound is not tight; but the gap with its true violation $C(\mathbf{x}_1, \mathbf{a}_1; \mathbf{w})$ is small, so the search in this interval may terminate. Note $(\mathbf{x}_1, \mathbf{a}_1)$ is indeed the MVC; but had $\sigma(f(\mathbf{x}_1, \mathbf{a}_1; \mathbf{w}))$ been to the left of the intersection point, then $(\mathbf{x}_2, \mathbf{a}_2)$ would have been the MVC despite the upper bound suggesting otherwise. In that case, the gap $U^*_u - C(\mathbf{x}_1, \mathbf{a}_1; \mathbf{w})$ would be larger, inducing a split of the interval.

The search for the MVC within $[f_\ell, f_u]$, as embodied in Eqs. (20–21), can be decomposed into a search over two disjoint classes of state-action pairs:

$$H^+_{\mathbf{w}} = \{(\mathbf{x}, \mathbf{a}) : h(\mathbf{x}, \mathbf{a}, \top; \mathbf{w}) \geq h(\mathbf{x}, \mathbf{a}, \bot; \mathbf{w})\} \quad (22)$$

$$H^-_{\mathbf{w}} = \{(\mathbf{x}, \mathbf{a}) : h(\mathbf{x}, \mathbf{a}, \top; \mathbf{w}) < h(\mathbf{x}, \mathbf{a}, \bot; \mathbf{w})\}. \quad (23)$$

By restricting the search to $H^+_{\mathbf{w}}$, and setting the constant probability term to σ_u, we obtain the BOP:

$$\max_{\mathbf{x},\mathbf{a}} \quad \sigma_u \cdot h(\mathbf{x}, \mathbf{a}, \top; \mathbf{w}) + (1 - \sigma_u) \cdot h(\mathbf{x}, \mathbf{a}, \bot; \mathbf{w}) \quad (24)$$

$$\textit{s.t.} \quad (\mathbf{x}, \mathbf{a}) \in H^+_{\mathbf{w}} \cap [f_\ell, f_u]. \quad (25)$$

A solution of this problem provides a maximizer $(\mathbf{x}^*_u, \mathbf{a}^*_u)$ with objective value U^*_u. Similarly, the optimization using $H^-_{\mathbf{w}}$ and the lower bound probability $\sigma_\ell = \sigma(f_\ell)$ provides an analogous solution $(\mathbf{x}^*_\ell, \mathbf{a}^*_\ell)$ with objective value U^*_ℓ.

Using the monotonicity of the objective w.r.t. σ, it can be shown that $U^* = \max(U^*_\ell, U^*_u)$ is an upper bound on the maximal constraint violation in $[f_\ell, f_u]$ (we denote by $MCV(I)$ the maximum *degree* of constraint violation in interval I). If the corresponding solution is $(\mathbf{x}^*, \mathbf{a}^*)$, the true violation $C(\mathbf{x}^*, \mathbf{a}^*; \mathbf{w})$ is a lower bound on $MCV([f_\ell, f_u])$. The gap $U^* - C(\mathbf{x}^*, \mathbf{a}^*; \mathbf{w})$ indicates how close it is to provably being the MVC within $[f_\ell, f_u]$.

This allows us to define conditions under which we can terminate the search: (i) if $U^* \leq \varepsilon$, for some small tolerance ε, there is no violation larger than ϵ in this interval; (ii) if $U^* \leq MCV(I)$, for some other interval I, the MVC (for the full subproblem) does not lie in this interval; (iii) if $U^* \leq C(\mathbf{x}^*, \mathbf{a}^*; \mathbf{w}) + \varepsilon$, then $(\mathbf{x}^*, \mathbf{a}^*)$ is (within ε of) the MVC in this interval. Should none of these conditions hold, we split the interval and recursively search in the two subintervals $[f_\ell, \frac{f_\ell + f_u}{2}]$ and $[\frac{f_\ell + f_u}{2}, f_u]$. When the subproblem search terminates with an MVC $(\mathbf{x}^*, \mathbf{a}^*)$, we add the *true constraint* to the master, i.e., $0 \geq C(\mathbf{x}^*, \mathbf{a}^*; \mathbf{w})$, not the approximate constraint using the approximated probability.

ALP-SEARCH will terminate with the identification of a violated constraint if one exists with degree at least ε. Thus, the final master LP may not include all violated constraints; but any unexpressed constraint is within ε of being satisfied. We analyze the error of this relaxation below. Furthermore, it will never generate a constraint that is not violated, so the master LP will never be overconstrained.

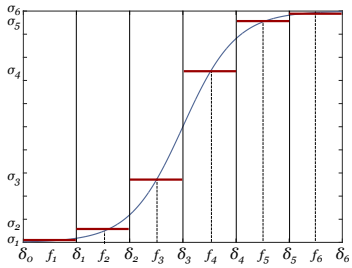

Figure 3: Sigmoid: a piecewise-constant approximation.

ALP-SEARCH can be modified easily to produce an *exact solution* to the subproblem using a simple *identical witness test*. When searching in interval $[f_\ell, f_u]$, the following test ensures the identified constraint is the true MVC in the interval: if the same state-action pair maximizes the degree of constraint violation using both constant probabilities σ_u and σ_ℓ, we can terminate with that constraint. This test can be implemented by solving two BOPs, over all $(\mathbf{x}, \mathbf{a}) \in [f_\ell, f_u]$, with constant probabilities σ_u and σ_ℓ, respectively.

Observation 1. *ALP-SEARCH with the identical witness test constructs an exact solution to the ALP optimization problem.*

As a result, an exact solution to for ALP-SEARCH is viable. However, we expect small approximation tolerances to encourage much faster convergence in practice.

A Piecewise Constant Approximation The CG procedure above provides an "exact" solution (modulo termination tolerance ε) to the subproblem by solving a *sequence* of BOPs—the formulation of one BOP depends on the results of a previous BOP. The same linearization, using a constant approximation for the (sigmoid) response probability over intervals $I = [f_\ell, f_u]$, can be applied non-adaptively. We can *fix* a set of intervals in f-space, and approximate the sigmoid in each interval with a constant σ_I. Such a *piecewise-constant (PWC)* approximation allows us to compute the MVC within each interval using a single BOP; and since the interval subproblems are independent, they can be solved in parallel. We call ALP using this means of constraint generation *ALP-APPROX*.

More precisely, we partition f-space into m intervals $[\delta_{i-1}, \delta_i]$, $1 \leq i \leq m$, and associate a constant response probability σ_i with $[\delta_{i-1}, \delta_i]$, where $\sigma_i = \sigma(f_i)$ for some $f_i \in [\delta_{i-1}, \delta_i]$; see Fig. 3. We require only that $\delta_0 \leq \min_{\mathbf{x},\mathbf{a}} f(\mathbf{x}, \mathbf{a})$ and $\delta_m \geq \max_{\mathbf{x},\mathbf{a}} f(\mathbf{x}, \mathbf{a})$. As above, the search for the MVC in $[\delta_{i-1}, \delta_i]$ is now linear:

$$\max_{\mathbf{x},\mathbf{a}} \quad \sigma_i h(\mathbf{x}, \mathbf{a}, \top; \mathbf{w}) + (1 - \sigma_i) h(\mathbf{x}, \mathbf{a}, \bot; \mathbf{w}) \quad (26)$$

$$\textit{s.t.} \quad \delta_{i-1} \leq f(\mathbf{x}, \mathbf{a}) \leq \delta_i. \quad (27)$$

Unlike ALP-SEARCH, we do not refine these intervals, but use a suitably chosen partitioning and fixed σ_i for each interval. We solve the subproblem for each interval I independently to obtain an approximate MVC $(\mathbf{x}^I, \mathbf{a}^I)$, compute the

true degree of violation $C(\mathbf{x}^I, \mathbf{a}^I; \mathbf{w})$ for each candidate, and return to the master the (true) constraint whose true violation is maximal. If the maximum is non-positive, we report no violated constraint and terminate the master.

This CG procedure generally incurs some error, since it may fail to detect a violated constraint for two reasons. First, the PWC approximation for some I may find no violations in I even though a violation exists whose degree has been underestimated. Second, while it may detect a violating state-action pair using the approximation, if the *true* violation value for this pair is non-positive, the procedure will claim no violation exists; yet a different state-action pair might exhibit a true violation in I. If this occurs in all intervals, no violated constraint will be added and the master will terminate.[8] We bound the error induced by such a failed detection below.

A key advantage of ALP-APPROX is that the subproblems for distinct intervals are independent and can be solved in parallel. This stands in contrast with exact ALP-SEARCH, where refined subproblems must be solved in sequence.

Approximation Error We first analyze the impact of ALP-APPROX (and by the same argument, ALP-SEARCH with tolerance ε) failing to detect a violated constraint by comparing solution quality with that of exact ALP. All proofs are provided in an extended version of the paper.[9]

Suppose ALP-APPROX terminates s.t. any missing constraint has violation of at most ϵ. We can show that: (a) the approximate VF produced by ALP-APPROX is no more than $\frac{\epsilon}{1-\gamma}$ away (in max-norm) from a feasible solution to (exact) ALP, i.e., it provides an upper bound on the ALP solution; and (b) the difference in our VF approximation (w.r.t. weighted 1-norm [de Farias and Van Roy, 2003]) is no more than $\frac{\epsilon}{1-\gamma}$. This holds since any infeasible ALP solution can be made feasible by increasing the bias weight. A bound on value loss of the induced policy is obtainable using methods of [de Farias and Van Roy, 2003].

We can determine how well the MDP induced by PWC discretization matches the true logistic MDP; i.e., how different the optimal VFs for the MDPs are when using the true and approximated dynamics, respectively. To obtain meaningful bounds, we assume that for any given policy π, the Markov process induced by applying π mixes with rate μ.[10] The maximum difference in VFs obeys:

$$|V_\pi - \widehat{V}_\pi|_\infty \leq \sqrt{\frac{(2\ln 2)}{\mu}\,\mathrm{LRE}(\widehat{\sigma})}\,\frac{1}{1-\gamma}\max_{\mathbf{x}} R(\mathbf{x}, \pi(\mathbf{x})),$$

where LRE, the log relative error, is defined as $\mathrm{LRE}(\widehat{\sigma}) = \ln \sup_f \max\left[\frac{\widehat{\sigma}(f)}{\sigma(f)}, \frac{1-\widehat{\sigma}(f)}{1-\sigma(f)}\right]$, and $\widehat{\sigma}$ is the PWC sigmoid approximation. Finally, we note that given a fixed discretization $[\delta_0, \ldots, \delta_k]$ of f-space, the optimal choice of constants minimizing LRE within an interval is $\widehat{\sigma}(f) = \ln \frac{\exp(\delta_i+\delta_{i+1})+\exp(\delta_{i+1})}{1+\exp(\delta_i)}$, for $\delta_i \leq f \leq \delta_{i+1}$. Moreover, a PWC approximation to the sigmoid achieving LRE of at most ϵ can be constructed with $O(\frac{1}{\epsilon}||\mathbf{u}||_1)$ intervals (see the extended paper for further details and proofs).

5 Empirical Results

We test the logistic MDP framework and the ALP-APPROX algorithm using models of various sizes derived from a targeted advertising domain. We generate logistic pCTR models that predict the probability of a user response φ, representing a click on a displayed (in-app) ad. pCTR models of several sizes are learned from data drawn over a period of one day with roughly 300M training examples. Each example has roughly 50 input random variables (with highly variable domain sizes), reflecting static user features, summaries of past behavior, ad characteristics, etc.

From these models, we construct a logistic MDP with a single binary response variable φ at the core of the logistic transition model. The transition functions are derived from the semantics of the features themselves. The reward function is 1 for a click, and our sequential objective is to maximize the expected discounted *cumulative click-through rate*.[11]

We produce four models:

- TINY has two (natural) state variables (with domain sizes 6 and 42) and one action variable (domain size 7), giving rise to a one-hot encoding with 55 *sparse binarized features (sbfs)*, of which 48 are state and 7 are action features. The induced state and action spaces have size 252 and 7, respectively.
- SMALL has: 6 state variables, 4 action variables; 86 sbfs (71 state, 15 action); 158,760 states, 126 actions.
- MEDIUM has: 11 state variables, 8 action variables; 421 sbfs (251 state, 170 action); approximately 2^{43} states, 220M actions.
- LARGE has: 12 state variables, 11 action variables; 2854 sbfs (2630 state, 224 action); approximately 2^{54} states, 2^{39} actions.

We run ALP-APPROX using one basis function per sbf (e.g., LARGE uses 2854 basis functions). We use GLOP[12] to solve the master LP, and SCIP[13] for the Boolean subproblems. We solve the subproblems using various numbers of subintervals, or *bands*, to assess the impact of coarse vs. fine discretization.

Fig. 4 shows, for TINY, SMALL and MEDIUM, how objective value improves (for both 25 and 100 bands) and maximum constraint violation (MVC) decreases (both estimated and true constraints at 100 bands) with CG iteration. In each case, ALP-APPROX converges very quickly given the number of potential constraints. We see little difference in objective performance between 25 and 100 bands: 100 converges to slightly higher values. Other results (not shown)

[8] If a constraint is missed in I during a round of CG, a violated constraint may still be found in I later (if CG continues), since the subproblem changes with each master iteration.

[9] Available at `research.google.com/pubs/CraigBoutilier.html`.

[10] See [Boyen and Koller, 1998] for the mixing concept used here; it has the desirable property that it can be bounded using the structure of the DBN.

[11] Including bid predictions in the model to optimize for social welfare or revenue is straightforward, as would be including other reward factors of interest, if contained within the features.

[12] See `https://developers.google.com/optimization/`.

[13] See `http://scip.zib.de/`.

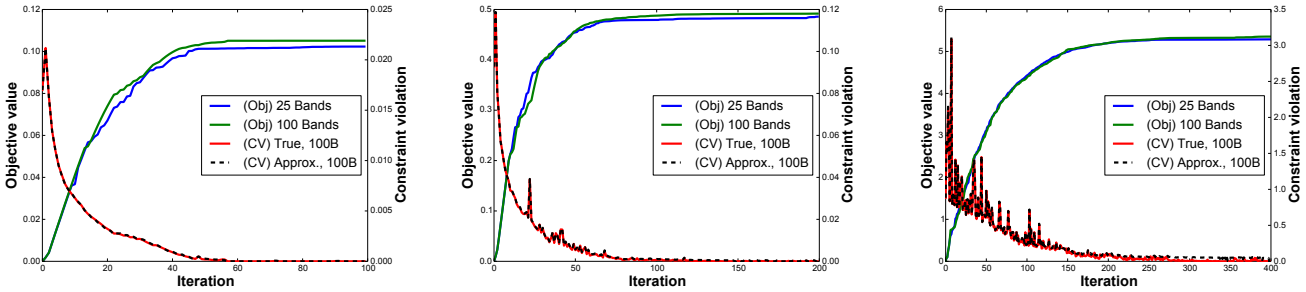

Figure 4: Objective value and max constraint violation progress as a function of the ALP constraint generation iteration for three MDP domains: TINY (left), SMALL (middle), MEDIUM (right).

provide no appreciable improvement with up to 400 bands (w.r.t. 100), and 50 bands seems to be the "sweet spot" for this domain. For TINY we computed the exact ALP solution (no sigmoid approximation) using CG: our approximation was within 99.96% of the exact solution with 100 bands (and 97.3% with 25 bands). For SMALL, exact CG was not feasible, but we used early termination to obtain a loose upper bound on the exact solution: our approximation (100 bands) was within 94.3% of the "exact" upper bound. (Exact solutions are not feasible for larger instances.)

Fig. 5 shows (non-distributed) computation time per iteration for MEDIUM (trends are similar in the other instances). Total time for 400 iterations with 25 bands is roughly 600s.: a distributed implementation would take roughly 24s. (since solution time is dominated by the subproblem BOPs).

Fig. 6 shows the metrics above for LARGE, using 50 bands and 3000 CG iterations. It has a similar anytime profile w.r.t. objective value as the smaller problems. Despite the tremendous size of this discrete MDP, a distributed implementation would complete 3000 iterations in an estimated 19 minutes.

Finally, we perform a simple empirical comparison of the cumulative reward generated by the MDP-optimized policy to that obtained using myopic optimization. Using the MEDIUM domain, we generate trials by sampling 50 random ads from the data set as the eligible action set,[14] a random initial state from the data set, and then measure reward accrued over trajectories of length 60 by the myopic and ALP policies. We sample 40 eligible sets and 2500 initial states for each set, giving 100K trials. The ALP policy shows a small improvement of about 0.52% in value over the myopic policy.[15]

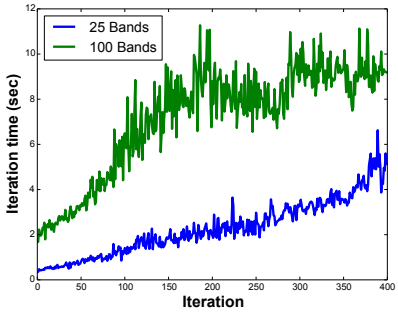

Figure 5: Computation time per CG iteration: MEDIUM.

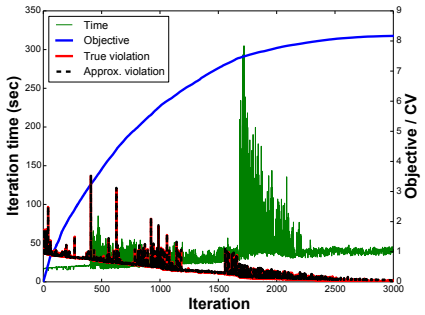

Figure 6: Objective value, max constraint violation, and computation time per CG iteration on the LARGE domain.

6 Conclusion and Future Work

We have proposed logistic MDPs as a model for sequential user interaction and developed approximate solution techniques based on ALP that handle the complexity introduced by logistic response variables. An attractive feature of Logistic MDPs is their exploitation of existing myopic response models and their features (the same applies to model-free RL), and our ALP method handles large problems but seems to approximate exact ALP very well.

We are exploring several extensions of our model (several of these extensions are described in the full paper). Multiple response variables are easily modeled in logistic MDPs, though they complicate the ALP method. Crosses of features are often used in logistic regression to handle nonlinearities, and are easily incorporated into ALP). More complex response models like deep neural networks can also be incorporated into logistic MDPs, and certain DNNs can be linearized to allow our ALP methods to be used.[16] We are also exploring the adaptation of these methods to nonlinear VF approximators. Finally, the idea of compiling LP constraints into a compact form [Guestrin *et al.*, 2003; Robbel *et al.*, 2016] rather than relying on constraint generation is an intriguing prospect.

[14] This reflects the fact that for any impression only a small set of ads are considered eligible for various reasons (e.g., targeting, budget throttling, campaign expiration, etc.).

[15] The feature subset used in this synthetic model gives a model with low pCTR and very limited user state, hence a small influence of actions on user state. Mean reward per trajectory: myopic: 0.75325; ALP: 0.75718; we did not find this difference to be statistically significant.

[16] Thanks to Ross Anderson for conversations on this topic.

References

[Amin *et al.*, 2012] K. Amin, M. Kearns, P. Key, A. Schwaighofer. Budget optimization for sponsored search: Censored learning in MDPs. *UAI-12*, 543–553, 2012.

[Archak *et al.*, 2010] N. Archak, V. Mirrokni, and S. Muthukrishnan. Mining advertiser-specific user behavior using adfactors. *WWW-10*, 31–40, 2010.

[Archak *et al.*, 2012] N. Archak, V. Mirrokni, and S. Muthukrishnan. Budget optimization for online campaigns with positive carryover effects. *WINE-12*, 86–99, 2012.

[Boutilier and Lu, 2016] C. Boutilier and T. Lu. Budget allocation using weakly coupled, constrained Markov decision processes. *UAI-16*, 52–61, 2016.

[Boutilier *et al.*, 1995] C. Boutilier, R. Dearden, and M. Goldszmidt. Exploiting structure in policy construction. *IJCAI-95*, 1104–1111, 1995.

[Boutilier *et al.*, 1999] C. Boutilier, T. Dean, and S. Hanks. Decision theoretic planning: Structural assumptions and computational leverage. *JAIR*, 11:1–94, 1999.

[Boyen and Koller, 1998] X. Boyen and D. Koller. Tractable inference for complex stochastic processes. *UAI-98*, 33–42, 1998.

[Charikar *et al.*, 1999] M. Charikar, R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. On targeting Markov segments. *STOC-99*, 99–108, 1999.

[Cheng *et al.*, 2016] H. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir, R. Anil, Z. Haque, L. Hong, V. Jain, X. Liu, H. Shah. Wide & deep learning for recommender systems. *Deep Learning for Rec. Sys.*, 7–10, 2016.

[Covington *et al.*, 2016] P. Covington, J. Adams, and E. Sargin. Deep neural networks for YouTube recommendations. *RecSys-16*, 191–198, 2016.

[Dean and Kanazawa, 1989] T. Dean and K. Kanazawa. A model for reasoning about persistence and causation. *Comput. Intell.*, 5(3):142–150, 1989.

[Graepel *et al.*, 2010] T. Graepel, J. Candela, T. Borchert, and R. Herbrich. Web-scale Bayesian click-through rate prediction for sponsored search advertising in Microsoft's Bing search engine. *ICML-10*, 13–20, 2010.

[Guestrin *et al.*, 2003] C. Guestrin, D. Koller, R. Parr, and S. Venkataraman. Efficient solution algorithms for factored MDPs. *JAIR*, 19:399–468, 2003.

[He and McAuley, 2016] R. He and J. McAuley. Fusing similarity models with Markov chains for sparse sequential recommendation. *ICDM-16*, 2016.

[Hoey *et al.*, 1999] J. Hoey, R. St-Aubin, A. Hu, and C. Boutilier. SPUDD: Stochastic planning using decision diagrams. *UAI-99*, 279–288, 1999.

[Hohnhold *et al.*, 2015] H. Hohnhold, D. O'Brien, and D. Tang. Focusing on the long-term: It's good for users and business. *KDD-15*, 1849–1858, 2015.

[Li *et al.*, 2009] T. Li, N. Liu, J. Yan, G. Wang, F. Bai, and Z. Chen. A Markov chain model for integrating behavioral targeting into contextual advertising. *Data Mining and Audience Intel. for Advert.*, 1–9, 2009.

[McMahan *et al.*, 2013] H. McMahan, G. Holt, D. Sculley, M. Young, D. Ebner, J. Grady, L. Nie, T. Phillips, E. Davydov, D. Golovin, S. Chikkerur, D. Liu, A. Wattenberg, M. Hrafnkelsson, T. Boulos, J. Kubica. Ad click prediction: A view from the trenches. *KDD*, 1222–1230, 2013.

[de Farias and Van Roy, 2003] D. P. de Farias, B. Van Roy. The linear programming approach to approximate dynamic programming. *Oper. Res.*, 51(6):850–865, 2003.

[Puterman, 1994] M. Puterman. *Markov Decision Processes: Discrete Stochastic Dyn. Prog.* Wiley, 1994.

[Rendle *et al.*, 2010] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme. Factorizing personalized Markov chains for next-basket recommendation. *WWW-10*, 811–820, 2010.

[Richardson *et al.*, 2007] M. Richardson, E. Dominowska, and R. Ragno. Predicting clicks: Estimating the click-through rate for new ads. *WWW-07*, 521–530, 2007.

[Robbel *et al.*, 2016] P. Robbel, F. Oliehoek, and M. Kochenderfer. Exploiting anonymity in approximate linear programming: Scaling to large multiagent mdps. *AAAI-16*, 2537–2543, 2016.

[Sahoo *et al.*, 2012] N. Sahoo, P. Singh, and T. Mukhopadhyay. A hidden Markov model for collaborative filtering. *Mgmt. Info. Sys. Qrtly.*, 36(4), 2012.

[Schuurmans and Patrascu, 2001] D. Schuurmans and R. Patrascu. Direct value approximation for factored MDPs. *NIPS-01*, 1579–1586, 2001.

[Shani *et al.*, 2005] G. Shani, D. Heckerman, and R. Brafman. An MDP-based recommender system. *JMLR*, 6:1265–1295, 2005.

[Silver *et al.*, 2013] D. Silver, L. Newnham, D. Barker, S. Weller, and J. McFall. Concurrent reinforcement learning from customer interactions. *ICML-13*, 924–932, 2013.

[St-Aubin *et al.*, 2000] R. St-Aubin, J. Hoey, C. Boutilier. APRICODD: Approximate policy construction using decision diagrams. *NIPS-00*, 1089–1095, 2000.

[Taghipour *et al.*, 2007] N. Taghipour, A. Kardan, and S. Ghidary. Usage-based web recommendations: A reinforcement learning approach. *RecSys-07*, 113–120, 2007.

[Tan *et al.*, 2016] Y. Tan, X. Xu, and Y. Liu. Improved recurrent neural networks for session-based recommendations. *Deep Learning for Rec. Sys.*, 17–22, 2016.

[Theocharous *et al.*, 2015] G. Theocharous, P. Thomas, and M. Ghavamzadeh. Personalized ad recommendation systems for life-time value optimization with guarantees. *IJCAI-15*, 1806–1812, 2015.

[Wu *et al.*, 2017] C. Wu, A. Ahmed, A. Beutel, A. Smola, and H. Jing. Recurrent recommender networks. *WSDM-17*, 2017.